1 Introduction

In its most general form, interconnection networks are a central component of all computing and communication systems from the internal interconnects of chip-scale embedded architectures to geographic-scale systems such as wide area networks and the internet. This section focuses on interconnection networks as they are used in multiprocessor and multicore systems. Specifically, the section addresses basic systems concepts and principles for interconnecting processors, cores, and memory modules. These concepts and principles are not exclusive to this application domain and in fact have also found application in networks used in storage systems, system area networks, and local area networks albeit with distinct engineering manifestations tailored to their particular domains.

Interconnection networks have their roots in telephone switching networks and packet switching data networks and in fact the earliest multiprocessor networks were adaptations of those networks. They have since evolved to incorporate new principles and concepts that accommodate the distinct engineering and performance constraints of multiprocessor and multicore architectures. In particular, the lack of standards such as that found in the wide-area and local-area communities created opportunities for meeting performance goals via customized solutions which in turn has been largely responsible for the diversity of implementations. However, the shared goal of high performance multiprocessor communication has led to a common set of principles and concepts organized around a network stack that can be viewed as being comprised of the following layers: routing layer, switching layer, flow control layer, and physical layer. These layers collectively operate to realize the network performance and correctness properties, e.g., deadlock freedom. The physical layer comprises signaling techniques for synchronized transfer of data across links while switching techniques and their interaction with the flow control layer are described in the section on switching techniques in this Encyclopedia. This section describes basic interconnection network concepts covering topology, routing, correctness, and common metrics. The discussion includes the routing layer and periodically exposes dependencies on the lower level layers particularly with
respect to deadlock freedom.

2 Topology

Routing of communication between nodes is the most basic property of an interconnection network and is intimately tied to the topology of interconnections between nodes where a node may be a processor, a core, a memory module, or an integrated processor-memory module. A node may a source or destination of a message that is encapsulated in one or more packets each of which is comprised of a header that contains routing and control information and an optional body that contains data. A crossbar switch provides a direct connection between any pair of network nodes. It is also the most expensive interconnect in terms of hardware and power. Thus, as system size grows we encounter a variety of topologies that represent various performance and cost tradeoffs. The most common topologies can be broadly classified into shared medium, direct, indirect, and hybrid.

2.1 Shared Medium Networks

The bus is a the most common shared medium network and the first widely used interconnect in multiprocessor architectures. A bus can have several hundred signals organized as address, data, and control. Only one node is allowed to transmit at a time as determined by the arbitration policy that arbitrates between concurrent requests for the bus.

Such shared medium buses do have several advantages for low node count systems. They provide efficient broadcast medium in support of one-to-all or one-to-many communication patterns. The shared bus also serves to serialize all traffic between nodes. When used as a processor-memory interconnect this property is exploited in cache-based multiprocessors to support coherence protocols for shared memory multiprocessors [13]. Routing on the bus is straightforward since all devices snoop the address lines. However if the bus is used to connect devices of differing speeds, such as processors to memory, bus utilizations can degrade rapidly if a transaction (i.e., reading from memory) holds the bus for the duration of the transaction. As processor memory speed disparities grew, low utilizations led to split phase transactions where requests and responses are split into two separate transactions. As a result multiple memory transactions can be interleaved on the bus increasing bus utilizations.

Physical constraints limit bus lengths and consequently system sizes. This has also become true for chip multiprocessors and thus the use of buses can be expected to be limited to smaller sized systems at future technology nodes.

2.2 Direct Networks

To achieve scalability beyond that feasible with shared medium networks, nodes are interconnected by point-to-point links. A node includes a router whose input and output links are connected to routers at some set of neighboring nodes producing a direct network. An example of a processor node in a two dimensional mesh network is illustrated in Figure 1 where one of the router links connects to the local node. Each unidirectional link realizes the injection and ejection channels.

The topology refers to the overall interconnection pattern between nodes (actually between the routers to which the node is connected). A common topology, particularly for on-chip networks, is a two dimensional mesh as shown in Figure 1. Formally, an n-dimensional mesh has $k_0 \times k_1 \times \cdots \times k_{n-2} \times k_{n-1}$ nodes, where $k_i$ is the number of nodes along dimension $i$ and is referred to as the radix for dimension $i$, and where $k_i \geq 2$ and $0 \leq i \leq n - 1$. Each node $X$ is identified by $n$
coordinates, \((x_{n-1}, x_{n-2}, \ldots, x_1, x_0)\), where \(0 \leq x_i \leq k_i - 1\) for \(0 \leq i \leq n - 1\). Two nodes are connected if their coordinates differ by 1 in only one dimension. Except for nodes on the boundary, all nodes are directly connected to \(2n\) other nodes. Thus we can have a 3x4 mesh where each interior node is connected to 4 other nodes or a 6x4x2 three dimensional mesh where each interior node is connected to 6 other nodes. Now messages between a pair of non-adjacent nodes are routed through the intermediate routers.

A symmetric generalization of the mesh is a \(k\)-ary \(n\)-cube \([4]\). Unlike the mesh, nodes at the edge have a link that “wraps around” the dimensions as shown in Figure 2 for the 2D torus. Now all nodes have the same number of neighbors. Formally, a \(k\)-ary \(n\)-cube has \(k_0 \times k_1 \times \cdots \times k_{n-2} \times k_{n-1}\) nodes, \(k_i\) nodes along each dimension \(i\) and and two nodes \(X\) and \(Y\) are neighbors if and only if \(y_i = x_i\) for all \(i\), \(0 \leq i \leq n - 1\), except for dimension \(j\), where \(y_j = (x_j \pm 1) \mod k\). Every node has \(2n\) neighbors. When \(n = 1\), the \(k\)-ary \(n\)-cube collapses to a bidirectional ring with \(k_0\) nodes. When the radix in all dimensions are equal to \(k\), we have \(k^n\) nodes. When \(k = 2\) the topology becomes the binary hypercube or binary \(n\)-cube as shown in Figure 2 which illustrates a 16-node binary hypercube.

A generalization of the \(k\)-ary \(n\)-cube is the generalized hypercube where rather than being connected to the immediate neighbors in each dimension, each node is connected to every node in each dimension. In graph theoretic terms, as we proceed from meshes through \(k\)-ary \(n\)-cubes to generalized hypercubes additional links are added creating a denser interconnect. From a practical standpoint as additional links are created, feasible channel widths change since the number of I/Os at an individual router have some maximum value. Finally, in all of the cases above, we note that the radix in each dimension can be distinct. These networks are often referred to as orthogonal networks recognizing the independence of the routing behavior in each dimension (see Section 5).

The landscape of direct interconnection network topologies is only limited by the imagination and many variants have been proposed and will no doubt continue to be proposed to match engineering constraints as technologies evolve. The preceding are the most common classes of networks used in current generation systems. Figure 2 illustrates some other common examples from the
current time frame. A two dimensional flattened butterfly topology used in on-chip networks is equivalent to a two dimensional generalized hypercube. Various forms of balanced and unbalanced trees have been proposed motivated by the logarithmic distance properties between nodes. Others such as de Bruijn networks [16], star networks [2] and cube connected cycles [15] present different tradeoffs between network diameter (longest path between nodes) and wiring density. We can anticipate other direct networks to evolve in the future.

A final observation is that all of the topologies in the preceding discussion exhibit some form of regularity in structure, the edge routers in the mesh notwithstanding. A final class of networks that can be distinguished are irregular networks. Rather than a mathematical description, such networks are typically represented as graphs of interconnections and can be constructed to form any type of topology. An example is shown in Figure 2. These networks are usually motivated by packaging constraints or customization, for example a specific topology in embedded systems where the topology is customized to the specific communication interactions between nodes. As a result of their irregularity these networks rely on distinct classes of routing techniques described in Section 3.
2.3 Indirect Networks

In general, direct topologies support interconnections where some routers may simply serve as connections to other routers and are not attached to node such as a processor, core or memory. These are the class of indirect networks. Like direct topologies, these may be organized into regular or irregular configurations.

A large class of regular indirect networks are multistage interconnection networks (MINs) an example of which is illustrated in Figure 3. Nodes are connected to the inputs and outputs of the network. Each 2x2 router - typically referred to as a switch in these topologies - can route a packet from one of its input ports to one of its output ports. There are \( N = 2^n \) input ports, \( N = 2^n \) output ports, and \( \log_2 N \) stages of switches with \( N/2 \) switches in each stage. The interconnections between stages of switches are important - they ensure that a single path exists between every input port and every output port. In particular consider the interconnection between stages 1 and 2 in Figure 3. The connections can be described by the function

\[
\sigma(x_{n-1}x_{n-2} \ldots x_1x_0) = x_{n-2} \ldots x_1x_0x_{n-1}
\]

For example, output 5 of stage 0 is connected to input 3 of the next stage of switches (as shown in the figure) and similarly output 6 is connected to input 5. This connection can be thought of as analogous to perfectly shuffling a deck of \( N \) cards, where the deck is cut into two halves and shuffled perfectly, i.e., the inputs are shuffled to the outputs. This function is referred to as the
shuffle function and the network in the figure is the well known Omega network. The preceding is a compact way of expressing the connections between stages. Note that every output (of a stage) is connected to exactly one input (of the next stage of switches) and therefore these connections can be represented as a permutation of the inputs. Consequently such networks are referred to as permutation networks. In general such networks can be constructed with $t \times t$ switches and correspondingly $n = \log_2 N$ stages for connecting $t^n$ nodes.

![Diagram of Omega network](image)

Figure 4: Some Example MINs

Not all stages need to have the same interconnect pattern. For example, consider the following interconnect function.

$$\beta_i(x_{n-1} \ldots x_{i+1}x_ix_{i-1} \ldots x_1x_0) = x_{n-1} \ldots x_{i+1}x_0x_{i-1} \ldots x_1x_i$$

The preceding function refers to Butterfly function forming the corresponding network of the same name illustrated in Figure 4(a). Note each stage uses a distinct butterfly ensuring every input is has a path to every output. Many such MINs have been defined based on different permutation functions (see [11] for a sampling). Perhaps the oldest and best known among these is the Clos Network [3]. The Clos network is derived from a recursive decomposition of the cross bar with the first level decomposition shown in Figure 5(a). A cross bar is decomposed into a 3 stage network. Consider a network with $N = r \times n$ inputs. The first stage comprises of $r, n \times m$ switches while the center stage comprises $m, r \times r$ switches. The last stage comprises $r, m \times n$ switches. Each switch at the first stage has a connection to every switch in the middle stage. Each switch in the center stage has a connection to every switch at the last stage. Thus there are $m$ paths between every input and output. Each of the individual switches can be further and recursively decomposed with $n = m = 2$ and $r = N/2$ until the network is reduced to $(2\log_2 N) - 1$ stages of $2 \times 2$ switches. This network is known as the well known Benes network while the Baseline network described above can be seen to form one half of this network.

The MINs are characterized based on their ability to simultaneously establish connections between inputs and outputs. If a path through the MIN prevents other paths between free inputs
and outputs from being set up, they are referred as blocking networks. MINs for which any free input can be connected to any free output are referred to as non-blocking while MINs where any free input can be connected to any free output by rearranging (re-routing) existing paths are referred as rearrangeable. The preceding MINs with \( \log_2 N \) stages are blocking. Note that adding extra stages introduces multiple paths between inputs and outputs until the network becomes rearrangeable and eventually non-blocking. Clos networks are non-blocking when \( m \geq 2n - 1 \) and rearrangeable when \( m \geq n \).

Finally, the preceding discussion was based on MINs with unidirectional links routing packets from input to output. However, MINs can operate with bidirectional links and packets can be routed to “any” output port of an intermediate switch leading to routing paths shown in Figure 4(b). Such MINs are referred to as bi-directional MINs. A careful look at the structure of bi-directional MINs will reveal that their exist multiple paths between any source-destination pair an example of which is shown in the figure.

A fat tree is an indirect network that is topologically a balanced binary tree topology with the modification that the bandwidth of the links increases as one gets closer to the root (hence the label “fat”) [14]. This is typically achieved by adding more links in parallel closer to the root. Routing is straightforward in a fat tree - a message is routed up to the nearest common ancestor and then down to the destination. Thus local communication is restricted to sub-trees. Fat trees are universal in the sense that it is the most efficient network for a given amount of hardware (see [14] for definitions of efficiency and universality). A butterfly BMIN with turnaround routing can be viewed as a fat tree - consider the dashed boxes of Figure 5(b) as single internal switches in a fat tree. In this case, it can be viewed as having four 2x2 switches in the first stage, the two 4x4 switches in the second stage and the single 8x8 switch in the last stage (the root of the tree). Modern fat tree topologies have evolved into a parametric family as a function of the number of levels in the tree, number of switches, at each level and the switch sizes.

As with direct networks, the preceding MINs are considered regular and it is possible to consider a range of irregular indirect networks of arbitrary topologies with nodes connected to an arbitrary
subset of switches. As before this has a clear impact on the routing protocols.

2.4 Hybrid Networks

As the name suggests, hybrid networks are customized networks that couple various topological concepts described so far. For example, we might have a mesh of clusters where nodes within a cluster are interconnected by a high speed bus. Or we may have a hierarchy of ring interconnections. Hybrid networks are often motivated by the target packaging environment and incorporation of new technologies. For example, continued progress in optical communication, especially silicon nanophotonics, will lead to networks where interconnects and topologies are optimized for a combination of optical and electrical signaling.

3 Routing

The routing algorithm establishes a path through the network and can be distinguished by the number of candidate paths that can be produced between a source and destination. Deterministic routing protocols provide a single path between every source destination pair. Oblivious routing algorithms produce routes independent of the state of the network, for example independent of local congestion. While deterministic routing algorithms are oblivious the converse may not be true. For example, the source may cycle through alternative (deterministic) paths to the destination for successive packets in an attempt to balance network traffic across all links while not being influenced by the current state of any router. Another example is Valiant’s two phase routing protocol [20] where a packet is first transmitted to a randomly chosen node and then routed deterministically to the destination. Adaptive routing algorithms operate in response to network conditions in making routing decisions at each router selecting from a set of alternative output ports at intermediate routers and thereby exposing multiple alternative paths to a destination. All of the preceding routing algorithms may cause a packet to follow a minimal path to the destination or permit non-minimal paths.

Implementations of routing algorithms admit to several alternatives. In distributed routing, each router implements a routing function that provides an output port(s) for each packet based on some combination of destination and/or input channel and/or source. Several implementations of distributed routing are feasible including table look-up or finite state machine implementations. In source routing the source node prepends an encoding of the entire path to the destination, for example as a sequence of output ports at each intermediate router. Each router along the path simply queries this header to determine the local output port along which to forward the packet. Source routing is invariably deterministic.

Many refined characterizations of routing algorithms have also emerged over the years. In multiphase routing algorithms, packets employ different routing algorithms during different phases of transmission. In optimistic routing algorithms, packets can freely choose any path to the destination since deadlock freedom is ensured via recovery rather than avoidance (see Section 6). Alternatively, in conservative routing algorithms, packets advance only when progress to the destination along the selected path is assured. Principle features of routing algorithms and an accompanying taxonomy can be found in [11].
4  Deadlock and Livelock

For any given topology the routing algorithms must realize deadlock-free and livelock free operation. Deadlock refers to the network state where some set of messages are indefinitely blocked waiting on resources held by each other. Livelock refers to a packet state, where a packet is routed through the network indefinitely and is never delivered. The presence of faulty components can lead to both deadlock and livelock behavior if mechanisms to prevent them are not implemented.

An example of a deadlocked configuration of messages is shown in Figure 6 where messages are being transmitted between sources $S_i$ and destinations $D_i$ in a 2D mesh using a deterministic shortest path routing and input buffered switches. Every message is waiting on a buffer used by another message producing a cycle of dependencies where each message is transitively waiting on itself. When routing is adaptive more complex configurations of deadlocked messages can occur where each message is blocked waiting on multiple buffers occupied by multiple messages and the set of messages cannot make progress since each message is transitively waiting on itself along all feasible output ports at the intermediate router.

Deadlock avoidance mechanisms are designed such that packets request and wait for resources (e.g. buffers) in a such a way as to avoid the formation of deadlocked configurations of messages or otherwise guarantee progress. Routing algorithms for a topology are designed in conjunction with switching techniques (described in a separate section in this Encyclopedia) to avoid deadlock. The most commonly employed principle, which is also common to deadlock avoidance in operating systems, is to provide a global ordering of resources and for all packets to request resources in this order. Specific applications of this principle are topology specific and described in Section 5.

In contrast, deadlock recovery mechanisms promote optimistic routing of messages absent any deadlock avoidance rules [1]. Deadlocked configurations of messages are detected and recovery mechanisms are invoked that re-establish forward progress of packets. The application of deadlock
recovery is motivated in network configurations where the probability of deadlock occurrence is very low. This is often the case in adaptively routed networks with large numbers of virtual channels. All deadlock freedom proofs rely on the consumption assumption - messages reaching a destination are eventually consumed. However, the endpoints can introduce dependencies between incoming packet channels and outgoing packet channels since there may be dependencies between message types, for example processing of a message may lead to the injection of new messages. Consequently it is possible to introduce a dependency between injection and ejection buffers external to the networks and which is not addressed by routing restrictions in the network. This can potentially introduce cyclic dependencies and therefore deadlock referred to as protocol or message-dependent deadlock [18]. For example, in shared memory systems message traffic consists of request packets (for cache lines) that produces response traffic (cache lines). Traffic is often separated into two virtual networks - the request network and response network. This protocol handlers at the end points can ensure that each network satisfies the consumption assumption and that the endpoint is never indefinitely blocked waiting on injection (full) or ejection (empty) channels.

5 Deadlock-Free Routing

The literature on packet data networks is rich with techniques for deadlock-free routing (avoidance). This section will be focused on principles and common techniques that are utilized in multiprocessor interconnection networks that in one form or another enforce ordering constraints to ensure deadlock freedom.

The simplest example of deadlock free routing is in 2D meshes using X-Y routing where a packet is routed in the X dimension eliminating the X-offset (find the right column) before routing in the Y dimension (eliminate the Y-offset) - dimension order routing. This avoids deadlocked configurations since no message traversing the Y dimension can request a channel in the X dimension and create a cycle of dependencies as illustrated in Figure 6. This deadlock-free routing algorithm is naturally extended to n dimensions where all packets follow the same order of dimensions in eliminating offsets in each dimension. The Turn Model generalizes dimension order restrictions on routing by observing that packets must be prevented from moving from one class of channels to another, i.e., performing a turn [12]. In dimension order routing in 2D meshes packets are prevented from turning from the Y dimension to the X dimension. Many Turn-based deterministic and adaptive routing algorithms have been developed in the literature. One of the principle attractions of the Turn model is that it can be applied to networks without virtual channels. Figure 7 shows all possible Turns for a 2D mesh and those pairs that can be restricted to prevent deadlock via X-Y routing. However, restricting the use of a single turn in each cycle is sufficient to avoid deadlock.

Deadlock free routing in irregular networks can be achieved with similar ordering constraints. For example, a distinguished node can be identified in the network (for example the root of a spanning tree). A labeling algorithm labels all channels are either being up and directed towards the root or down and being directed away from the root. Packet routing is now constrained to use
all up channels followed by all down channels. A packet cannot turn from down channel to an up channel avoiding the creation of cyclic dependencies [17].

The wrap-around links in multidimensional tori introduce hardware cycles within a dimension and therefore dimension order routing is not deadlock free (consider the case where each node in a row is transmitting to a destination $h$-hops away in the same row). For deterministic routing the work of Dally and Seitz [6] showed that the necessary and sufficient condition for deadlock-free routing is an acyclic channel dependency graph - intuitively if a packet can be routed from channel $A$ to channel $B$, there is a dependency from $A$ to $B$. Acyclic dependency graphs are achieved in tori with the presence of two virtual channels per physical channel - VC0 and VC1. A single link in each dimension is chosen (sometimes called the dateline link) and packets being routed across this link are restricted from 'turning” from one virtual channel class to the other. Packets not crossing the dateline link use a single virtual channel class. With the preceding restriction routing in the network is deadlock-free. The requirement of acyclic channel dependency graphs can be implemented in many ways by creatively restricting the use of virtual channels in regular and irregular networks.

The theory of deadlock free adaptive routing protocols was developed by Duato [7, 8]. Intuitively cycles may exist in the channel dependency graph as long as there exists an acyclic subgraph that enables all messages to be routed between any source destination pair - sometimes referred to as an escape path or escape channel at a router. Virtual channels provide a natural medium for implementing adaptive routing. In the case of tori, each physical link supports three virtual channels - VC0, VC1, and VC2. The first two provide the acyclic subgraph necessary for deadlock free routing. VC2 is the adaptive channel that can be used by a packet to cross any physical link to the destination. The theory has been developed for wormhole, packet and virtual cut-through implementations [7, 10, 11]. In general, by suitably specifying routing restrictions, adaptive routing protocols can be developed for a regular and irregular networks.

6 Metrics

The construction of an interconnection network is a process of tradeoffs between various network design parameters such as topology, channel widths, and switch degree and the physical constraints of the target implementation environment such as on-chip vs. in-rack vs. inter-rack. The consequences of the choice of network parameters are captured in several common metrics.

A practical constraint encountered by interconnection networks is the wiring density. Since networks must be implemented in three dimensions, for an $N$ node network the available wiring space can be seen as growing as $O(N^{2.5})$ which corresponds to the area of the surface of a 3D cube containing the $N$ nodes. Network traffic can grow as $O(N)$. As system sizes are scaled wiring limits are eventually encountered. A common metric to capture wiring limits is the bisection width of a network defined as the minimum number of wires that must be cut when the network is divided into two equal sets of nodes. The intuition is that in the worst case, every node on one side of the minimum cut will be communicating with every node on the other side of the cut maximally stressing the communication bandwidth of the network. For regular topologies such as $k$-ary $n$-cubes the minimum bisection width is orthogonal to the dimension with the largest radix. For irregular topologies the min-cut defines the bisection width. Bisection bandwidth is usually computed as the bandwidth across these wires implicitly considering only data signals. The bisection width of several topologies where each link between adjacent node is $W$ bits wide are listed in Table 1.

A second physical constraint is the number of data signals on a router referred to as the node degree, node size, or pin-out. The importance of node degree stems from the fact that it is a
practical limit on routers. Further, the topology, bisection width, and node degree are closely related. Consider the implementation of a $k$-ary $n$-cube with the bisection width constrained to be $N$ bits. The maximum channel width determined by this bisection width is $W$ bits and given by

$$\text{Bisection width} = 2Wk^{n-1} = N \Rightarrow W = \frac{k}{2}$$

The channel width directly impacts message latency and hence performance. The specific relationships between channel widths, node degree, and bisection bandwidth vary significantly whether the network is implemented on-chip, intra-rack or inter-rack. These metrics also provide a uniform basis for comparison between various topologies.

7 Bibliographic Notes and Further Reading

Each of the major topics discussed in this section have been subjected to a rich and diverse development. A good coverage of fundamental architectural, theoretical, and system concepts and a distillation of key concepts can be found in two texts [5, 11]. Advanced treatments can be found in papers in most major systems and computer architecture conferences.

References


Table 1: Examples of bisection width and node size of some common networks ($t \times t$ switches are assumed in the MIN).

<table>
<thead>
<tr>
<th>Network</th>
<th>Bisection Width</th>
<th>Node Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-ary $n$-cube</td>
<td>$2Wk^{n-1}$</td>
<td>$2Wn$</td>
</tr>
<tr>
<td>Binary $n$-cube</td>
<td>$\frac{WN}{2}$</td>
<td>$nW$</td>
</tr>
<tr>
<td>$n$-dimensional mesh</td>
<td>$Wk^{n-1}$</td>
<td>$2Wn$</td>
</tr>
<tr>
<td>MIN</td>
<td>$NW$</td>
<td>$2tW$</td>
</tr>
</tbody>
</table>


